

Brief Papers

Extended Input Space Support Vector Machine

Ricardo Santiago-Mozos, *Member, IEEE*,
Fernando Pérez-Cruz, *Senior Member, IEEE*, and
Antonio Artés-Rodríguez, *Senior Member, IEEE*

Abstract—In some applications, the probability of error of a given classifier is too high for its practical application, but we are allowed to gather more independent test samples from the same class to reduce the probability of error of the final decision. From the point of view of hypothesis testing, the solution is given by the Neyman–Pearson lemma. However, there is no equivalent result to the Neyman–Pearson lemma when the likelihoods are unknown, and we are given a training dataset. In this brief, we explore two alternatives. First, we combine the soft (probabilistic) outputs of a given classifier to produce a consensus labeling for K test samples. In the second approach, we build a new classifier that directly computes the label for K test samples. For this second approach, we need to define an extended input space training set and incorporate the known symmetries in the classifier. This latter approach gives more accurate results, as it only requires an accurate classification boundary, while the former needs an accurate posterior probability estimate for the whole input space. We illustrate our results with well-known databases.

Index Terms—Classifier output combination, multiple sample classification, Neyman–Pearson, support vector machines.

I. INTRODUCTION

We are given a set of K samples: $\mathcal{T} = \{\mathbf{x}_1^*, \dots, \mathbf{x}_K^*\}$, where $\mathbf{x}_j^* \in \mathbb{R}^d$, and we are told that all of them belong to one of two possible alternatives. If the competing hypotheses are represented by their density functions, respectively, $p_1(\mathbf{x})$ and $p_{-1}(\mathbf{x})$, the most powerful test is given by the Neyman–Pearson lemma, which compares the product of the likelihood ratios for each \mathbf{x}_j^* to a threshold, which is determined by the size of the test [1]. The *Type II* error of the test decreases exponentially with K and the error exponent is given by the

Kullback–Leibler divergence between $p_1(\mathbf{x})$ and $p_{-1}(\mathbf{x})$ [2]. Therefore, the Neyman–Pearson lemma provides a tradeoff between the number of samples that we take before deciding and the achievable probability of error.

This problem is standard in many applications of simple hypothesis testing. For example, in radar [3], several samples are collected prior to declaring whether a target is present. In medical applications, a subject is tested several times before a disease can be diagnosed, because some tests are unreliable and present high false-positive and misdetection rates. A neuron collects several spikes [4] before it detects a certain pattern. Taking a decision with several samples allows the reduction of the probability of error (misdetections and false alarms) at the cost of gathering more information and/or waiting longer. In all these applications, the samples are known to come from the same class and are gathered to increase reliability.

In classification problems, the likelihoods $p_1(\mathbf{x})$ and $p_{-1}(\mathbf{x})$ are unknown and we are given a set of samples (i.e., the training set) that describes each likelihood. If $p_1(\mathbf{x})$ and $p_{-1}(\mathbf{x})$ are known to belong to a parametric family, we could estimate those parameters and apply the likelihood ratio to decide the best hypothesis. Nevertheless, for these estimates the properties described by the Neyman–Pearson lemma do not hold. And more often than not, $p_1(\mathbf{x})$ and $p_{-1}(\mathbf{x})$ cannot be described by known parametric distributions and we would need to resort to nonparametric estimation methods. In any case, if we want to classify \mathcal{T} into two alternative hypothesis, we would be ill-advised to estimate $\hat{p}_1(\mathbf{x})$ and $\hat{p}_{-1}(\mathbf{x})$ and apply a product of likelihood ratio test (because it is an ill-posed problem [5]) instead of directly building a classifier from the training set, which assigns a label to our test data.

When the likelihoods are unknown and we are given a training dataset, there is no equivalent result to the Neyman–Pearson lemma, which tells us how to take a unique decision for K test samples that are known to come from the same class, because they have been gathered that way. Two possible alternatives come to mind. First, train any classifier and combine its outputs for each of the K test samples to come up with a unique decision for these samples. We refer to this solution as the *consensus decision* and we explore it in Section II of this brief.

Second, we propose to build a classifier that directly classifies the K test samples belonging to the same class. This direct option works with an extended input space that takes K samples at once and trains the desired classifier. In order to do this, we need to transform the original d -dimensional input space into a Kd -dimensional space that represents the same problem and build the classifier with it. We refer to this solution as the *direct decision* and we explore it in Section III.

Manuscript received September 14, 2009; revised July 29, 2010 and October 26, 2010; accepted October 27, 2010. This work was supported in part by Ministerio de Educación of Spain under projects DEIPRO TEC2009-14504-C02-01 and COMONSENS CSD2008-00010. F. Pérez-Cruz was supported in part by Marie Curie Fellowship 040883-AI-COM. R. Santiago-Mozos has been supported in part by Marie Curie Transfer of Knowledge Fellowship of the EU sixth Framework Programme under contract CT-2005-029611 and Fundación Española para la Ciencia y la Tecnología, Ministerio de Educación of Spain.

R. Santiago-Mozos is with the College of Engineering and Informatics, National University of Ireland Galway, Galway, Ireland (e-mail: ricardo.santiago-mozos@nuigalway.ie; frsmozos@tsc.uc3m.es.edu).

F. Pérez-Cruz and A. Artés-Rodríguez are with the Signal Theory and Communication Department, Universidad Carlos III de Madrid, Madrid 28903, Spain (e-mail: fernando@tsc.uc3m.es; antonio@tsc.uc3m.es).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TNN.2010.2090668

We need to exploit the structure embedded in our problem to obtain a reduced complexity and high-performance classifier. The set \mathcal{T} of test samples are not ordered and any order should provide the same outcome, as they all come from the same class. To exploit this symmetry in the extended input space, we have decided to adapt a support vector machine (SVM) [6] to this framework. SVMs are state-of-the-art nonlinear versatile classifiers that are well known and easy to train, although our procedure also holds for most classification algorithms of interest. We refer to our extended input space SVM as ESVM, which takes the set \mathcal{T} and gives a binary decision.

The consensus and direct decisions present different pros and cons, which make each technique desirable in different scenarios. The consensus decision needs accurate posterior probability estimates in the whole range (which might be hard to come by for high confidence decisions), however, it only needs to work with an input space of dimension d . The direct decision, on the one hand, operates with an input space of dimension Kd that has been extended artificially without having more training samples to build the classifier, so it might be more exposed to the curse of dimensionality. On the other hand, the direct decision only needs to return a decision, not a posterior probability estimate. Also the consensus decision can be used for any value of K , while in the direct decision K is specific. It is not clear cut which algorithm would be preferable, although in the experiments carried out it seems that the direct approach might be preferable (lower probability of error).

The rest of this brief is organized as follows. In Section II, we establish how to combine the posterior probability estimates from each test sample to reach a consensus label. We present the direct decision in Section III and the extended input space SVM in Section IV. We introduce an illustrative example in Section V together with the performance of the proposed extended input space SVM with well-known databases. We conclude in Section VI.

II. CONSENSUS DECISION

We want to classify \mathcal{T} into two possible alternatives and we have a soft-output classifier whose output can be interpreted as a posterior probability estimate, i.e.,

$$p(y^*|\mathbf{x}^*) \quad y^* \in \{\pm 1\}. \quad (1)$$

We could be tempted to compute the posterior probability estimate as

$$p(y^* = 1|\mathcal{T}) = \frac{\prod_{k=1}^K p(y_k^* = 1|x_k^*)}{\prod_{k=1}^K p(y_k^* = 1|x_k^*) + \prod_{k=1}^K p(y_k^* = -1|x_k^*)} \quad (2)$$

and decide accordingly. But (2) relies on each of the test samples being independent, but they are not, because all the test samples belong to the same class. To assign a consensus label to the test set with K samples we proceed

as follows:

$$\begin{aligned} p(y^* = 1|\mathcal{T}) &= \frac{p(y^* = 1)p(\mathbf{x}_1^*, \dots, \mathbf{x}_K^*|y^* = 1)}{p(\mathbf{x}_1^*, \dots, \mathbf{x}_K^*)} \\ &= \frac{p(y^* = 1) \prod_k p(\mathbf{x}_k^*|y^* = 1)}{p(\mathbf{x}_1^*, \dots, \mathbf{x}_K^*)} \\ &= \frac{p(y^* = 1) \prod_k \frac{p(y^* = 1|x_k^*)p(\mathbf{x}_k^*)}{p(y^* = -1|x_k^*)}}{p(\mathbf{x}_1^*, \dots, \mathbf{x}_K^*)} \\ &= \frac{\prod_k p(y^* = 1|x_k^*)}{p(y^* = 1)^{K-1}} \cdot (3) \\ &= \frac{\prod_k p(y^* = 1|x_k^*)}{p(y^* = 1)^{K-1} + \prod_k p(y^* = -1|x_k^*)} \end{aligned}$$

To obtain the second equality, we have applied the fact that the K test samples are independent given the label, and for the third equality we have applied Bayes rule, as we did for the first.

To decide for either hypothesis, we need to multiply the posterior probability estimate for each sample and divide by the prior probabilities to the $K-1$ power. We assign the consensus label to the K test sample set by the following rule:

$$y^* = \begin{cases} 1, & \frac{\prod_k p(y^* = 1|x_k^*)}{p(y^* = 1)^{K-1}} > \frac{\prod_k p(y^* = -1|x_k^*)}{p(y^* = -1)^{K-1}} \\ -1, & \text{otherwise.} \end{cases} \quad (4)$$

III. DIRECT DECISION

In this section, we describe an algorithm for extending the training set to a Kd -dimensional space, in which we can classify the K test samples directly. Given a training set $\mathcal{D} = \{\mathbf{x}_1^1, \dots, \mathbf{x}_{n_+}^1, \mathbf{x}_1^{-1}, \dots, \mathbf{x}_{n_-}^{-1}\}$, where $\mathbf{x}_i^c \in \mathbb{R}^d$ and $c \in \{\pm 1\}$ denotes the class label, we define the initial extended input space training set for class +1 as

$$\mathbf{Z}_o^1 = [\mathbf{z}_1^1 \quad \mathbf{z}_2^1 \quad \dots \quad \mathbf{z}_{\bar{n}_+}^1] = \begin{bmatrix} \mathbf{x}_{\ell_{11}}^1 & \mathbf{x}_{\ell_{12}}^1 & \dots & \mathbf{x}_{\ell_{1\bar{n}_+}}^1 \\ \mathbf{x}_{\ell_{21}}^1 & \mathbf{x}_{\ell_{22}}^1 & \dots & \mathbf{x}_{\ell_{2\bar{n}_+}}^1 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_{\ell_{K1}}^1 & \mathbf{x}_{\ell_{K2}}^1 & \dots & \mathbf{x}_{\ell_{K\bar{n}_+}}^1 \end{bmatrix} \quad (5)$$

which is a $Kd \times \bar{n}_+$ matrix containing in its columns the samples of the extended input space with $\bar{n}_+ = \lfloor n_+/K \rfloor$. The indices ℓ_{ij} are samples without replacement for $1, \dots, n_+$, consequently, in each column of \mathbf{Z}_o^1 we have an independent representation of the training set. We can similarly obtain an extended input space training set for class -1, namely \mathbf{Z}_o^{-1} .

Building the training set this way presents a severe limitation, as it reduces the number of training samples by a factor of K and makes the direct classifier harder to train. But there are symmetries in this problem that can be readily included in the training set that increases the number of training samples. These symmetries additionally impose constraints in the optimization problem, which simplifies its training procedure, as we explain in the next section. Let us use a representative example to illustrate this point. Suppose that $K = 2$ and $n_+ = 10$, a possible \mathbf{Z}_o^1 might be

$$\mathbf{Z}_o^1 = [\mathbf{z}_1^1 \quad \mathbf{z}_2^1 \quad \mathbf{z}_3^1 \quad \mathbf{z}_4^1 \quad \mathbf{z}_5^1] = \begin{bmatrix} \mathbf{x}_1^1 & \mathbf{x}_4^1 & \mathbf{x}_{10}^1 & \mathbf{x}_8^1 & \mathbf{x}_6^1 \\ \mathbf{x}_7^1 & \mathbf{x}_2^1 & \mathbf{x}_5^1 & \mathbf{x}_3^1 & \mathbf{x}_9^1 \end{bmatrix}. \quad (6)$$

For example, in this set we have the extended sample $\mathbf{z}_1^1 = \begin{bmatrix} \mathbf{x}_1^1 \\ \mathbf{x}_7^1 \end{bmatrix}$, and we should expect the sample $\begin{bmatrix} \mathbf{x}_7^1 \\ \mathbf{x}_1^1 \end{bmatrix}$ to present the same label, because the ordering of the sample should not matter. Therefore, we can extend the initial training set by including this permutation

$$\mathbf{Z}_p^1 = \begin{bmatrix} \mathbf{z}_{11}^1 & \mathbf{z}_{21}^1 & \mathbf{z}_{31}^1 & \mathbf{z}_{41}^1 & \mathbf{z}_{51}^1 & \mathbf{z}_{12}^1 & \mathbf{z}_{22}^1 & \mathbf{z}_{32}^1 & \mathbf{z}_{42}^1 & \mathbf{z}_{52}^1 \\ \mathbf{x}_1^1 & \mathbf{x}_4^1 & \mathbf{x}_{10}^1 & \mathbf{x}_8^1 & \mathbf{x}_6^1 & \mathbf{x}_7^1 & \mathbf{x}_2^1 & \mathbf{x}_5^1 & \mathbf{x}_3^1 & \mathbf{x}_9^1 \\ \mathbf{x}_7^1 & \mathbf{x}_2^1 & \mathbf{x}_5^1 & \mathbf{x}_3^1 & \mathbf{x}_9^1 & \mathbf{x}_1^1 & \mathbf{x}_4^1 & \mathbf{x}_{10}^1 & \mathbf{x}_8^1 & \mathbf{x}_6^1 \end{bmatrix}. \quad (7)$$

In this notation, the first subindex in $\mathbf{z}_{i\ell}$ identifies the sample, and the second the permutation.

This procedure can be applied for any number of training samples n_+ and n_- and any number of test samples K . For any K , \mathbf{Z}_p^1 , and \mathbf{Z}_p^{-1} , respectively, contain $\bar{n}_+K!$ and $\bar{n}_-K!$ columns of dimension Kd , as we need to incorporate all possible permutations \mathbf{z}_i^1 .

Finally, to further increase the sample diversity of the training procedure, we randomly create R samples for \mathbf{Z}_o^1 and \mathbf{Z}_o^{-1} from \mathcal{D} . Then, we build the final training matrix for the direct classifier as follows:

$$\mathbf{Z} = [\mathbf{Z}^1 \mid \mathbf{Z}^{-1}] = [\mathbf{Z}_{p1}^1 \mid \mathbf{Z}_{p2}^1 \mid \dots \mid \mathbf{Z}_{pR}^1 \mid \mathbf{Z}_{p1}^{-1} \mid \mathbf{Z}_{p2}^{-1} \mid \dots \mid \mathbf{Z}_{pR}^{-1}]. \quad (8)$$

IV. EXTENDED INPUT SPACE SVM

Now we can use the matrix \mathbf{Z} in (8) to train any binary nonlinear classifier of our liking, and we can then classify K test samples directly. But by doing so, we are ignoring the symmetries that we have included in this extended training set, from which the learning machine should benefit. Adding these symmetries to the learning procedure would be classifier dependent. Therefore, we operate from now on with SVMs, although, as mentioned earlier, similar steps can be applied to most classifiers of interests. We have chosen SVMs, because they are state-of-the-art learning machines that can be easily trained and have been used in many different applications with considerable success. In what follows, we assume the reader is already familiar with soft-margin nonlinear SVMs and its primal and dual representations, in any case, a detailed presentation can be found in [6], [7].

The SVM primal optimization functional solves

$$\min_{\mathbf{w}, b, \xi_i} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i \quad (9)$$

subject to

$$y_i \left(\mathbf{w}^\top \sum_{\ell} \phi(\mathbf{z}_{i\ell}) + b \right) \geq 1 - \xi_i \quad (10)$$

$$\xi_i \geq 0 \quad (11)$$

where we have removed the superindex in $\mathbf{z}_{i\ell}$, which determines the class label, and have replaced it by y_i , which also takes the values $+1$ or -1 , depending on the class label.¹ In (10) and (11), $i = 1, \dots, N$, with $N = R(\bar{n}_+ + \bar{n}_-)$ and

$\ell = 1, \dots, K!$ contains all the possible permutations for any training sample. The function $\phi(\cdot)$, which can be characterized by its kernel $k_\phi(\cdot, \cdot) = \phi(\cdot)^\top \phi(\cdot)$, is a nonlinear mapping of the input to a higher dimensional space [6].

In the standard SVM formulation, the slack variables ξ_i and the class label y_i would also be indexed by ℓ , because there can be a nonzero slack for each training sample, and each sample has its own class label. But by construction, we have included all the symmetries in the training set, so y_i is identical for any ℓ and $\mathbf{w}^\top \phi(\mathbf{z}_{i\ell})$ is independent of ℓ . To clearly understand this last point, see the example of a training set in (7), in which the training set presents all permutations of each training sample, consequently, given the symmetries, the learning machine has no information to provide us with different outputs for each permutation.

The Lagrangian for this problem becomes

$$\mathcal{L}(\mathbf{w}, b, \xi_i, \alpha_i, \eta_i) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i - \sum_i \eta_i \xi_i - \sum_i \alpha_i \left(y_i \left(\mathbf{w}^\top \sum_{\ell} \phi(\mathbf{z}_{i\ell}) + b \right) - 1 + \xi_i \right) \quad (12)$$

which has to be minimized with respect to the primal variables (\mathbf{w} , b and ξ_i) and maximized with respect to the dual variables (α_i and η_i). We can now compute the Karush–Kuhn–Tucker conditions [8]

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i y_i \phi(\mathbf{z}_i) = \mathbf{0} \quad (13)$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_i \alpha_i y_i = 0 \quad (14)$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = C - \eta_i - \alpha_i = 0 \quad (15)$$

where we have defined

$$\phi(\mathbf{z}_i) = \sum_{\ell} \phi(\mathbf{z}_{i\ell}) \quad (16)$$

with \mathbf{z}_i being any $\mathbf{z}_{i\ell}$.

We can use standard optimization tools to obtain the dual formulation

$$\max_{\alpha_i} \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j k_\phi(\mathbf{z}_i, \mathbf{z}_j) - \sum_{i=1}^N \alpha_i \quad (17)$$

subject to (14) and $0 \leq \alpha_i \leq C$. The kernel is given by

$$\begin{aligned} k_\phi(\mathbf{z}_i, \mathbf{z}_j) &= \phi(\mathbf{z}_i)^\top \phi(\mathbf{z}_j) = \sum_{\ell_1=1}^{K!} \sum_{\ell_2=1}^{K!} \phi(\mathbf{z}_{i\ell_1})^\top \phi(\mathbf{z}_{j\ell_2}) \\ &= \sum_{\ell_1=1}^{K!} \sum_{\ell_2=1}^{K!} k_\phi(\mathbf{z}_{i\ell_1}, \mathbf{z}_{j\ell_2}) = K! \sum_{\ell=1}^{K!} k_\phi(\mathbf{z}_i, \mathbf{z}_{j\ell}). \end{aligned} \quad (18)$$

The solution for this nonlinear transformation is quite interesting, because it adds the symmetry in the dual, we are training with all the possible symmetries of a training sample without needing to add a support vector for each new symmetric sample and we use a single nonlinear transformation.

¹We have modified the notation in this section to make it compatible with standard SVM notation, as it makes this section easier to understand.

For large K , the computational complexity can be quite high and we use LibSVM [9] to train the SVM. There are other approaches that can train the SVM with over a million training samples [10], [11] and that can also be applied to solve the ESVM.

V. EXPERIMENTS

We first show a toy 1-D example, in which we illustrate the benefits of classifying K samples directly, instead of combining the individual decisions. We then move on to classify different well-known databases, which allows the drawing of some general conclusions about the performance of our extended input space SVM.

In the figures and tables, we denote as SVM^K the solution reached by consensus, combining K SVM outputs. To obtain the SVM^K solution, we have used (4) and have transformed the SVM soft outputs into posterior probability estimates using Platt's method [12]. We use LibSVM [9] to train the SVM, and the implementation of Platt's method is given in [13]. We denote as ESVM^K the solution of the extended input space SVM with K samples. We also use LibSVM to train the ESVM.

In all the experiments, the hyperparameters of the SVM and ESVM have been computed by cross-validation and we have used a radial basis function kernel. We have set $N = Kn$ (i.e., $R = N/(\bar{n}_+ + \bar{n}_-)$), where n is the number of training samples in the original database. We have found empirically that increasing the number of training samples makes the ESVM^K predictions more stable. In any case, the information in the training sets for the SVM and ESVM is the same.

A. Toy Example

We are given n samples generated from a zero-mean unit-variance Gaussian that describes the class +1, and for the class -1 we are also given n samples from a unit-mean Gaussian with variance 4. We first train a nonlinear SVM with 50 samples from each class. We show in Fig. 1(a) the posterior estimate given by the SVM using Platt's method (dashed line) and the true posterior probability (solid line). As we mentioned in the introduction, we notice that the SVM accurately predicts if a sample is correctly classified, because, if we threshold the decisions at 0.5 in Fig. 1(a), we can see that the SVM predictions and the true posterior decisions almost coincide. But the SVM posterior probability estimates are far from accurate when we are away from the classification boundary and it does not accurately estimate extreme posterior probabilities [e.g., $x > 4$ or $x < -4$ in Fig. 1(a)].

We have plotted in Fig. 1(b) the Bayes (solid), the SVM^2 (dashed), and the ESVM^2 (dash-dotted) decision boundaries. We see that the ESVM^2 is closer to the optimal decision function and it does not have the artifacts that the SVM^2 presents, due to its inaccurate posterior probability estimates.

In Fig. 2(a) and (b), we show the probability of error as a function of K for 20 and 50 training samples per class, respectively. (For this experiment, we did not impose the symmetries in the kernel and we only train with the

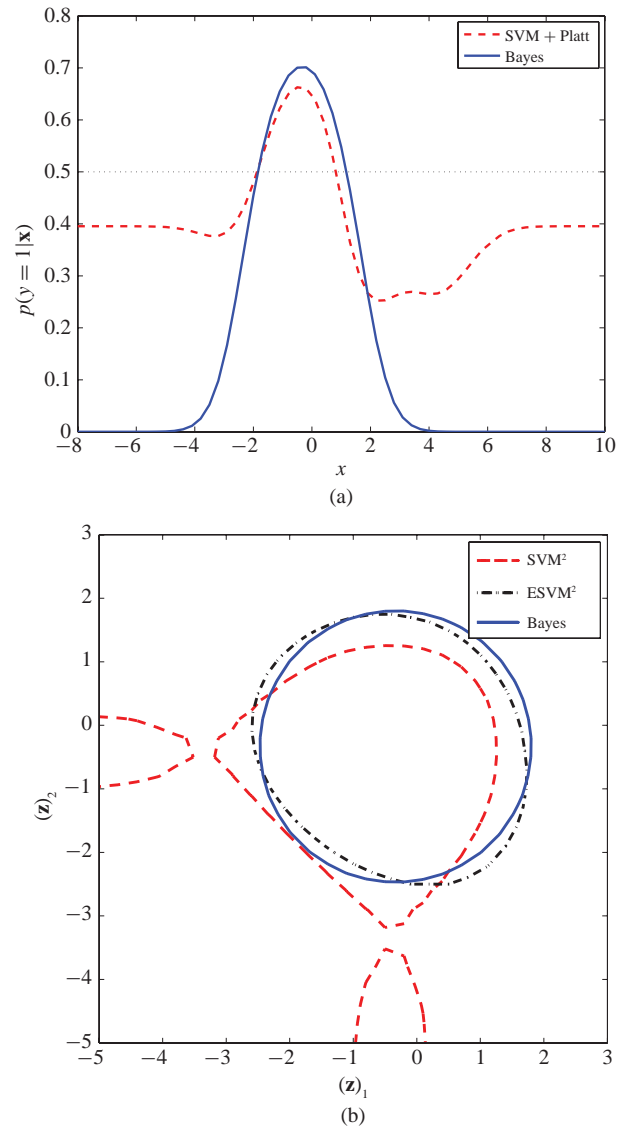


Fig. 1. (a) True posterior probability and its estimate using an SVM with 50 training samples and Platt's method. (b) Optimal decision boundary together with the SVM^2 and ESVM^2 classification functions.

extended input space training sets \mathbf{Z}_o^1 and \mathbf{Z}_o^{-1} , as $K!$ grows exponentially with K .) To obtain these plots, we have run 10^4 independent trials with 10^5 test samples. The probability of error reduces as we increase K . In the figures, we also see that the performance gap between the SVM^K and the ESVM^K increases with K . This is an expected result, because as K increases the inaccuracies in the SVM posterior probability estimate are more noticeable, when we compute the consensus label for the test set. The ESVM^K only focuses on the decision boundary and it does not need to give accurate soft outputs. It is also noteworthy that the probability of error reduces even when K becomes larger than the number of training samples per class. It can be seen that for larger K , the ESVM^K and SVM^K solution are better and closer to each other and to the Bayes solution, but still the ESVM^K outperforms the SVM^K for all K .

Finally, in Table I, we show the performance of the SVM^2 and ESVM^2 as we increase the number of training samples

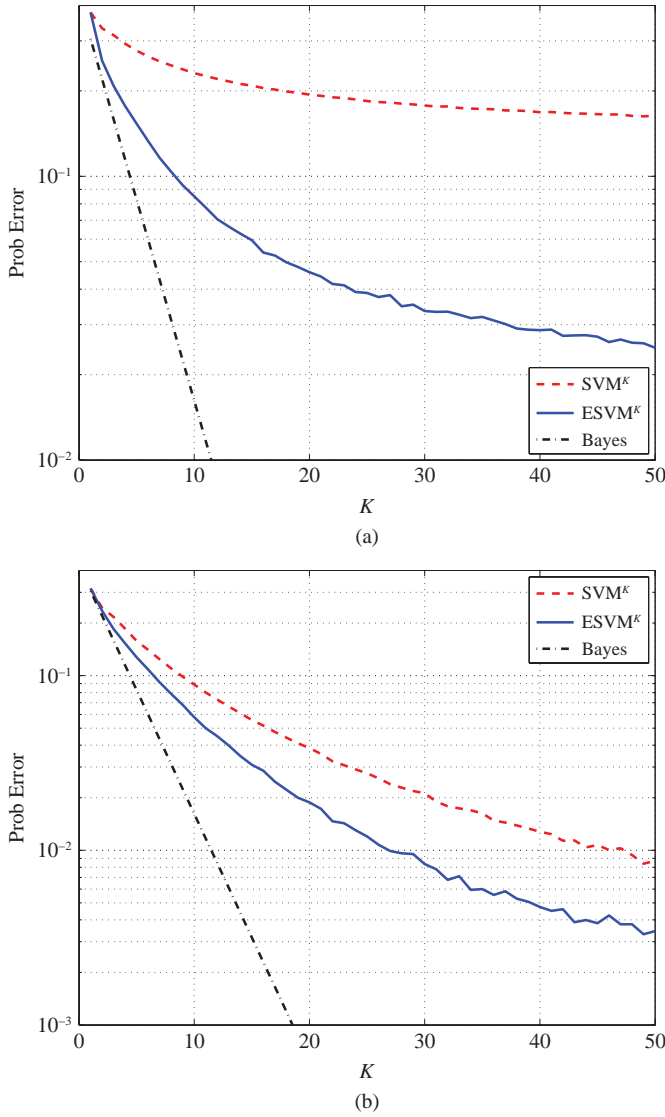


Fig. 2. Error rate for the toy example as a function of K for (a) 40 and (b) 100 training samples.

TABLE I
PROBABILITY OF ERROR OF SVM^2 AND $ESVM^2$, AS THE NUMBER OF TRAINING SAMPLES PER CLASS IS INCREASED TOGETHER WITH THEIR STANDARD DEVIATION

# samples	20	50	200
SVM^2	0.333 ± 0.154	0.249 ± 0.030	0.240 ± 0.020
$ESVM^2$	0.264 ± 0.040	0.237 ± 0.016	0.225 ± 0.006

per class. Both methods converge to the same solution, which corresponds to the Bayes classifier whose error is 0.22 for this problem with $K = 2$. The results in this table have been obtained with 10^4 independent trials with 10^5 test samples.

B. Real Databases

We have also carried out experiments with the 13 databases in [14]. Each database has been preprocessed to present zero mean and unit variance, and 100 training and test samples sets have been generated, except for *Splice* and *Image*, which only use 20. In Table II, we present the databases and some of

TABLE II
SUMMARY OF THE 13 DATABASES USED IN THIS SECOND EXPERIMENT: ITS NAME, DIMENSION, NUMBER OF TRAINING AND TEST PATTERNS, AND THE SVM PREDICTION ERROR

Name	Dim	# Train	# Test	SVM
Titanic	3	150	2051	$2.28e-1 \pm 1.2e-2$
Flare-solar	9	666	400	$3.23e-1 \pm 1.8e-2$
Banana	2	400	4900	$1.09e-1 \pm 5.6e-3$
Breast-cancer	9	200	77	$2.52e-1 \pm 4.5e-2$
Diabetes	8	468	300	$2.32e-1 \pm 1.7e-2$
Waveform	21	400	4600	$9.80e-2 \pm 4.4e-3$
Ringnorm	20	400	7000	$1.50e-2 \pm 9.5e-4$
Twonorm	20	400	7000	$2.43e-2 \pm 1.4e-3$
Thyroid	5	140	75	$4.62e-2 \pm 2.1e-2$
German	20	700	300	$2.41e-1 \pm 2.2e-2$
Heart	13	170	100	$1.55e-1 \pm 3.4e-2$
Splice	60	1000	2175	$1.08e-1 \pm 7.4e-3$
Image	18	1300	1010	$3.24e-2 \pm 6.1e-3$

TABLE III
 SVM^3 SOLUTION COMPARED WITH THE $ESVM^3$ FOR 13 DATABASES. THE $HSVM^3$ SOLUTION SHOWS THE SVM^3 PERFORMANCE WITH HARD OUTPUTS

Name	SVM^3	$ESVM^3$	$HSVM^3$
Titanic	$1.85e-1 \pm 2.2e-2$	$1.36e-1 \pm 2.3e-2$	$1.88e-1 \pm 2.3e-2$
Flare-solar	$2.31e-1 \pm 3.7e-2$	$1.87e-1 \pm 3.0e-2$	$2.46e-1 \pm 3.8e-2$
Banana	$1.78e-2 \pm 3.9e-3$	$1.53e-2 \pm 3.3e-3$	$3.53e-2 \pm 4.3e-3$
Breast-cancer	$1.89e-1 \pm 6.8e-2$	$1.80e-1 \pm 6.5e-2$	$2.46e-1 \pm 6.0e-2$
Diabetes	$1.23e-1 \pm 3.1e-2$	$1.18e-1 \pm 2.8e-2$	$1.74e-1 \pm 3.1e-2$
Waveform	$1.15e-2 \pm 2.9e-3$	$1.04e-2 \pm 2.8e-3$	$3.05e-2 \pm 5.0e-3$
Ringnorm	$3.52e-4 \pm 4.7e-4$	$5.14e-5 \pm 1.4e-4$	$8.75e-4 \pm 5.8e-4$
Twonorm	$3.64e-4 \pm 3.6e-4$	$3.04e-4 \pm 3.1e-4$	$1.83e-3 \pm 7.9e-4$
Thyroid	$4.00e-4 \pm 4.0e-3$	$4.17e-4 \pm 4.2e-3$	$1.03e-2 \pm 1.8e-2$
German	$1.46e-1 \pm 3.1e-2$	$1.47e-1 \pm 3.4e-2$	$1.99e-1 \pm 3.7e-2$
Heart	$5.28e-2 \pm 3.7e-2$	$5.80e-2 \pm 4.0e-2$	$7.13e-2 \pm 4.0e-2$
Splice	$1.65e-2 \pm 5.0e-3$	$2.04e-2 \pm 4.5e-3$	$3.17e-2 \pm 6.7e-3$
Image	$1.04e-3 \pm 1.7e-3$	$4.17e-3 \pm 4.6e-3$	$2.68e-3 \pm 2.9e-3$

their key features, together with the best SVM solution. For all the experiments in this section, we have set $K = 3$, although the results can be readily extended for larger values of K . To build the test set for the extended input space with $K = 3$, we first split the test database into two parts, one for the class +1 samples and the other for the class -1 samples. We then take three consecutive examples from each part without replacement until all the samples have been used. To compute the prior probabilities for the consensus decision, we use the relative frequencies in the training set.

In Table III, we report the probability of error for SVM^3 and $ESVM^3$. In Table IV, we compare with two statistics the errors in Table III to measure the difference between SVM^3 and $ESVM^3$ and report whether these differences are statistically significant. The first one is the classic *t-test* [15] and the second one is a more conservative *corrected resampled t-test* [16]. We have used boldface to denote that $ESVM^3$ is better than SVM^3 and we have used italic-face when SVM^3 is better than $ESVM^3$. For the *t-test*, there are seven databases in which $ESVM^3$ is superior to SVM^3 and four otherwise. For the more conservative test, only six databases pass the statistically significant threshold. This is an expected result, as the $ESVM$

TABLE IV

TWO STATISTICS COMPUTED TO COMPARE WHETHER THE DIFFERENCE BETWEEN ESVM^K AND SVM^K ARE SIGNIFICANT. BOLD FACED VALUES INDICATES ESVM^K IS SUPERIOR AND ITALICS ARE USED OTHERWISE

Name	t -Test in [15]	Test in [16]
Titanic	5.26e-200	5.30e-101
Flare-solar	2.23e-195	2.00e-096
Banana	7.25e-073	1.84e-006
Breast-cancer	3.31e-127	1.83e-033
Diabetes	4.49e-102	5.28e-017
Waveform	1.35e-040	0.0279
Ringnorm	1.80e-008	0.543
Twonorm	0.224	0.903
Thyroid	0.730	0.973
German	<i>3.21e-037</i>	<i>0.0451</i>
Heart	<i>9.58e-104</i>	<i>6.90e-018</i>
Splice	<i>7.53e-019</i>	<i>2.62e-007</i>
Image	<i>4.57e-017</i>	<i>5.55e-006</i>

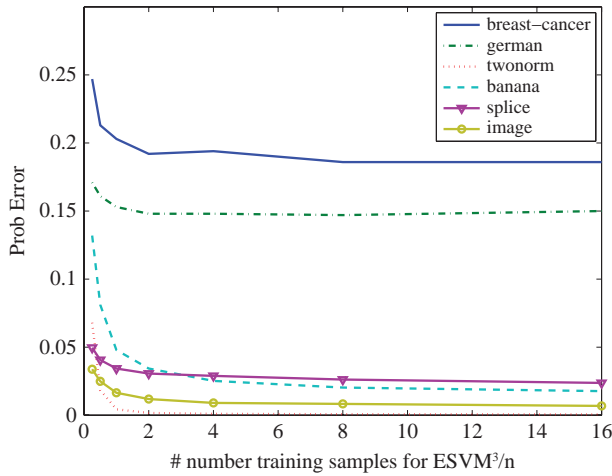


Fig. 3. Probability of error as we artificially increase the extended input space training set for six representative databases.

has some limitations and it should not always be better than SVM^K , but it is clear that in some cases it is much better and for some problems might be the way to improve the solution as we gather more examples.

In the Table III, we also report the SVM^3 performance with hard outputs, denoted by HSVM^3 . These results show that, even though Platt's method is inaccurate for short training sequences, it is better than not using the SVM soft output at all. Also, if we compare the results for the SVM in Table II and the ESVM^3 or SVM^3 in Table III, we can see a significant gain in all cases. Either of the proposed methods would improve the performance of the SVM, if we can gather more independent samples.

Finally, we show the probability of error of six representative databases for ESVM^3 in Fig. 3 when we increase the

number of training samples of the extended input space. We have generated training sets with $0.25n$, $0.5n$, n , $2n$, $4n$, $8n$, and $16n$, where n is the number of training patterns in Table II. We notice that once we use n training samples, there is little improvement, as the amount of information to learn the classifier is limited by n , not the number of repetitions that we use.

VI. CONCLUSION

When the likelihoods are unknown and we are given a training dataset, there is no equivalent result to the Neyman–Pearson lemma, which tells us how to take a unique decision for K test samples. We explored two alternatives to solve this problem. The consensus decision takes the posterior probability estimates to predict a single label for the set \mathcal{T} , and the direct decision builds a classifier that classifies \mathcal{T} in one take. We have also shown how the symmetries of the extended input space can be added to SVMs to give more accurate and reduced complexity classifiers.

REFERENCES

- [1] L. Wasserman, *All of Statistics*. New York: Springer-Verlag, 2004.
- [2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.
- [3] H. V. Poor, *An Introduction to Signal Detection and Estimation*, 2nd ed. New York: Springer-Verlag, 1994.
- [4] P. Dayan and L. F. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. Cambridge, MA: MIT Press, 2005.
- [5] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.
- [6] B. Schölkopf and A. Smola, *Learning with Kernels*. Cambridge, MA: MIT Press, 2001.
- [7] F. Perez-Cruz and O. Bousquet, "Kernel methods and their potential use in signal processing," *IEEE Signal Process. Mag.*, vol. 21, no. 3, pp. 57–65, May 2004.
- [8] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. New York: Wiley, 1987.
- [9] R.-E. Fan, P.-H. Chen, and C.-J. Lin, "Working set selection using the second order information for training SVM," *J. Mach. Learn. Res.*, vol. 6, pp. 1889–1918, Dec. 2005.
- [10] T. Joachims, "Training linear SVMs in linear time," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Philadelphia, PA, 2006, pp. 217–226.
- [11] S. S. Keerthi and D. DeCoste, "A modified finite Newton method for fast solution of large scale linear SVMs," *J. Mach. Learn. Res.*, vol. 6, pp. 341–361, Dec. 2005.
- [12] J. C. Platt, "Probabilities for SV machines," in *Advances in Large Margin Classifiers*, A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, Eds. Cambridge, MA: MIT Press, 2000, pp. 61–73.
- [13] H. Lin, C.-J. Lin, and R. C. Weng, "A note on Platt's probabilistic outputs for support vector machines," *Mach. Learn.*, vol. 68, no. 3, pp. 267–276, Oct. 2007.
- [14] G. Rätsch, B. Schölkopf, A. J. Smola, S. Mika, T. Onoda, and K.-R. Müller, "Robust ensemble learning," in *Advances in Large Margin Classifiers*, A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, Eds. Cambridge, MA: MIT Press, 2000, pp. 207–220.
- [15] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. San Mateo, CA: Morgan Kaufmann, 2005.
- [16] C. Nadeau and Y. Bengio, "Inference for the generalization error," *Mach. Learn.*, vol. 52, no. 3, pp. 239–281, Sep. 2003.